

Designing Real-World Embedded PowerPC Applications

You can use the Xilinx Embedded Development Kit to interface custom IP cores to the PowerPC processor.

by Eric Lynum
Hardware Engineer
Apptive Technologies
elynum@apptivetech.com

Developing a complete processor system for a real-world application can be a daunting task for first-time users. Given today's aggressive design requirements and challenges, you might not have the time to learn a new tool and adequately use the design flow needed to wield it. This can lead to costly mistakes.

Fortunately, Xilinx offers a design methodology that can effectively eliminate these problems. By demonstrating design techniques and implementing an application-specific PowerPC™ system, I'll show you how to utilize Xilinx® Embedded Development Kit (EDK) tools.

Introduction to the Design Example

A custom design by Apptive Technologies for one of our customers demonstrates how to use the high-performance architecture of Xilinx Platform Studio (XPS).

We helped our customer develop an electromagnetic sensor design that would provide high-resolution imaging of sub-surface objects by utilizing capacitive and inductive sensor arrays to detect and identify objects with the aid of applied electric and magnetic fields. Multiple frequency measurements in the hertz to megahertz range using inductive and capacitive sens-

ing techniques provide new characterization capabilities for hidden objects. A scientific tool called a dielectrometer can detect and identify sub-surface objects when the dielectric properties vary with frequency. Technology such as this provides basic sub-surface mapping, including core sampling, tunneling, drilling, digging, or locating water on Mars or other planetary bodies.

By using XPS, we created a completely reprogrammable digital interface to the analog circuitry required to connect to a sensor

operation of the direct digital synthesizer (DDS) to sweep over the range of frequencies from 1 Hz to 1 MHz and interface to analog circuitry. Using a Xilinx ML403 development board, the device targeted is a Virtex™-4 XC4VFX12 FPGA. The FPGA implements the main functionality of the numerically controlled oscillator and its interconnection with the internal Xilinx digital-to-analog converter (DAC). In addition, a custom daughtercard interfaces to the ML403. This gives the system the added bonus of

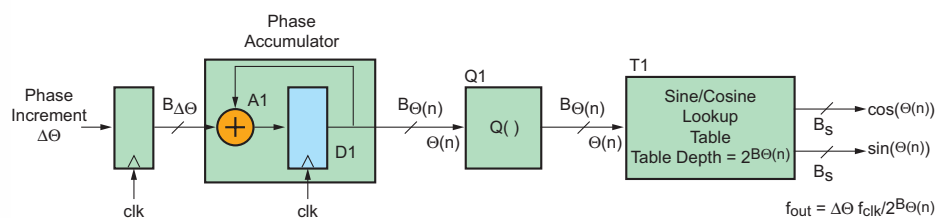


Figure 1 – DDS core design block diagram

for a dielectrometer system. Our completely reconfigurable platform has fewer components, making the system less expensive with a simpler, flexible topology.

Apptive's example shows the integration of XPS and ISE™ development tools. The design is a multi-wavelength dielectrometer system (MWDS) developed to implement and demonstrate the

directly controlling the analog components without needing extra parts.

Through a configurable user interface, the PowerPC sends commands to control the DDS frequency. The DDS core comprises a FIFO along with an internal ROM look-up table (LUT). A 100-MHz clock frequency is supplied to the DDS. The PowerPC reads and writes the 28-bit

data signal and 5-bit address inputs to the direct memory access (DMA) using the on-chip peripheral bus (OPB). Allowing user-configurable parameters definitely aided us in our design; if we needed to change any parameter on the fly to tailor them to a different set of requirements, we could do so quickly and easily.

Figure 1 illustrates the major components of the DDS core. The address provides time-multiplexed channel operation and the 28 bits of data are used in an algorithm to calculate the phase increment along with the 100-MHz clock.

Figure 2 depicts the FPGA core design. The DMA engine on the OPB sends the data from the FIFO to the internal FPGA DAC. One 32-bit analog control register off of the OPB individually operates the switches of a daughtercard containing the analog circuitry that plugs into the ML403 development board. The serial peripheral interface (SPI) sends serial commands to operate the analog multiplexers.

Figure 3 shows a block diagram of the MWDS.

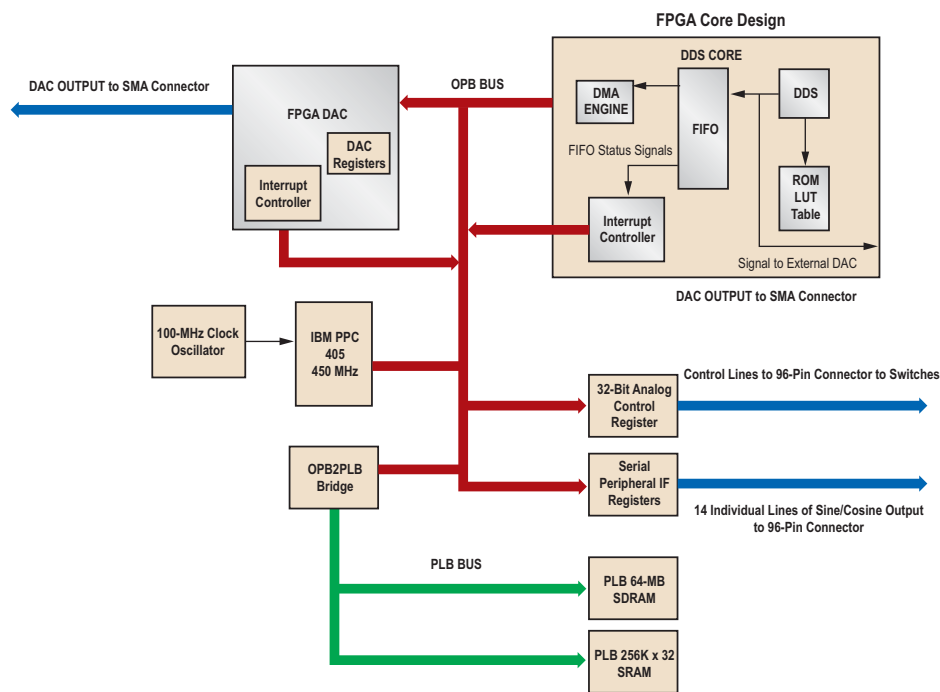


Figure 2 – FPGA core design block diagram

Design Example Modules

The main technical blocks of the MWDS perform these functions:

- A Virtex-4 FPGA provides a DDS core for generating a real- or complex-valued sinusoid, employing a LUT scheme that stores samples of that sinusoid. These samples are then presented to the onboard FPGA DAC or to the daughtercard DAC.
- The ML403 development board contains a DAC on-chip peripheral core that converts a binary number into an analog voltage output directly proportional to the value of the binary number. Either the FPGA DAC or the daughtercard can supply the analog output required. The output from either this DAC or the FPGA DAC is presented to the quad single-pole double-throw (SPDT) analog switches.
- The DMA engine block provides a single physical channel of direct memory access between the DDS and the internal DAC.

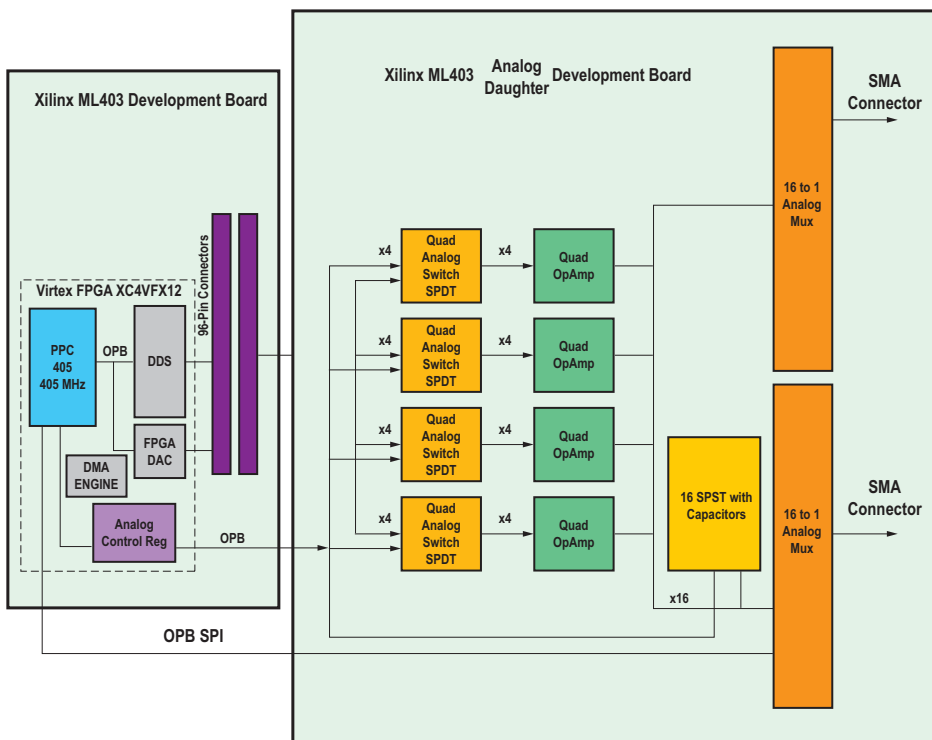


Figure 3 – Multi-wavelength dielectrometer system block diagram

- Four SPDT switch packages reside on the ML403 daughtercard and provide four selectable switches within one integrated circuit (IC) package, for a total of 16 for the PowerPC processor to turn on and off individually.
- Four quad single-pole single-throw (SPST) switch packages reside on the ML403 daughtercard and provide four selectable switches within each integrated circuit (IC) package, for a total of 16 for the PowerPC processor to turn on and off individually.
- The daughtercard uses two Analog Devices ADG725 CMOS dual 16-channel analog multiplexers with a three-wire control interface programmed by the PowerPC processor. The ADG725 switches one of 16 inputs to a common output that connects to a sub-miniature version A (SMA) connector. The analog multiplexers will interface to the PowerPC through a serial peripheral interface (SPI).

Design Tools

To implement this design methodology, you will need to use these tools:

- Xilinx Platform Studio (XPS), part of the Embedded Development Kit (EDK), includes a GUI and all tools run by the GUI to process hardware and software system components. You will use XPS to design your system interface. You can also perform system verification within the XPS environment.
- ISE software implements designs onto Xilinx programmable logic devices. XPS depends on ISE components to synthesize the microprocessor hardware design, map it to an FPGA target, and generate and download the bitstream. You must use the same version of EDK and ISE software to implement your designs.
- The IBM CoreConnect tool kit is not included with EDK. You will have to register with Xilinx to obtain the CoreConnect license. This tool performs bus functional simulation and verification of your system.

- Xilinx CORE Generator™ software imports parameterized cores of ready-made functions for use with Xilinx FPGAs.
- The Create and Import Peripheral Wizard creates custom peripherals and imports them into XPS. EDK uses the intellectual property interface (IPIF) library to implement custom processor peripherals along with a bus protocol called IP Interconnect (IPIC).

Implementing the Design

As you can see, Xilinx supplies a comprehensive set of software tools with which to implement designs. To use them in our multi-wavelength dielectrometer system, we followed these steps:

1. Generate a new XPS project using the Base System Builder. In this project, we initially created the PowerPC, RS232 serial interface, flash and OPB DAC, and SPI controller cores.
2. Create the DDS core using CORE Generator software parameterized for a 100-MHz DDS clock rate, an output width of 12 bits, a data and accumulator width of 28 bits, a phase angle of 10 bits, and a latency of 3 clock cycles. This generates .vhd, .edif, and .ngc files, which go into the XPS project directory.
3. Use the Create and Import Peripheral Wizard (in create mode) to generate HDL templates, the bus functional model (BFM) simulation platform, and ISE support files and software driver templates under the XPS project directory. As you go through the GUI menus, you will choose the OPB interface, number of interrupts, number of software registers, and DMA engine. This creates two files: a user logic file and a top-level VHDL file.
4. Use ISE software to implement custom functionality. Navigate to the ISE project file generated by the Create and Import Peripheral Wizard and open the .npl file. Then instantiate your FIFO core into the user logic section of the VHDL file.

5. Synthesize your ISE project file to verify that there are no errors.
6. In the XPS directory, go to the BFL (bus functional language) directory under bus functional simulation (BFSIM) and change the BFL scripts to the appropriate settings for testing the core. To compile BFL scripts, call up a DOS window and go to the pcores directory (where BFL scripts are located) and type the following command: `xilbfc <your file name>.bfl`.
7. Import the code to EDK by opening up the Create and Import Peripheral Wizard (import mode) and following the steps in the GUI.
8. Open the XPS project and add the custom DMA core. The custom core should appear along with the processor, RS232, SDRAM, and flash controller cores.
9. Place and route the design using XPS. Within XPS, you can use the GUI to download the bit file, where the design can run in real time.

Conclusion

XPS, used in combination with ISE software, creates a complete solution for implementing a PowerPC system. Our particular application demonstrates the benefits of using these tools, including reconfigurability, minimum component usage, and flexible configuration methodologies.

In this article, I have also shown how to tailor these tools for your specific application. For new users, this is important because you need to gain an understanding of the design flow and the procedures required to implement your designs accurately and in a timely manner. Following a proven design methodology and using the required tools can give you the confidence to achieve successful results in your design applications. Building on this foundation for designs targeting even the Virtex-5 multi-platform FPGA should yield a productive design.

If you have any questions or suggestions, e-mail elynum@apptivetech.com. 